

**IN THE UNITED STATES PATENT AND TRADEMARK OFFICE  
BEFORE THE BOARD OF PATENT APPEALS AND INTERFERENCES**

Appellants:	Novaes et al.	Group Art Unit: 2155
Serial No.:	09/584,259	Examiner: Michael Young Won
Filed:	05/31/00	Appeal No.:
Title:	METHOD, SYSTEM AND PROGRAM PRODUCTS FOR MANAGING PROCESSING GROUPS OF A DISTRIBUTED COMPUTING ENVIRONMENT	

To: Mail Stop Appeal Briefs – Patents  
Commissioner for Patents  
P.O. Box 1450, Alexandria, VA 22313-1450

**Brief of Appellants**

Dear Sir:

This is an appeal from a final rejection mailed October 3, 2006, rejecting claims 50-53, 56, 57, 59, 61-63, 66-69, 72-74, 76-81, 84-86 & 88-90, all the pending claims of the above-identified application. This Brief is accompanied by a transmittal letter authorizing the charging of appellants' deposit account for payment of the requisite fee set forth in 37 C.F.R. §41.20(b)(2).

Appellants' Brief is believed in compliance with the requirements set forth in 37 CFR §41.37(c). However, if Appellants' Brief does not comply with the requirements set forth in 37 CFR §41.37(c), Appellants request notification of the reasons for noncompliance and the opportunity to file an amended Brief pursuant to 37 CFR §41.37(d).

### **Real Party in Interest**

This application is assigned to **International Business Machines Corporation** by virtue of an assignment executed by the co-inventors on September 6, 2000, September 7, 2000, September 22, 2000, and September 25, 2000; and recorded with the United States Patent and Trademark Office at reel **011185**, frame **0283**, on September 29, 2000. Therefore, the real party in interest is **International Business Machines Corporation**.

### **Related Appeals and Interferences**

To the knowledge of the Appellants, Appellants' undersigned legal representative, and the assignee, there are no other appeals or interferences, which will directly affect or be directly affected by or have a bearing on the Board's decision in the instant appeal.

### **Status of Claims**

This patent application was filed on May 31, 2000 with the United States Patent and Trademark Office. As filed, the application included six (6) claims, three (3) of which were independent claims (i.e., claims 1, 3, 5).

A preliminary amendment was filed on April 12, 2002 adding claims 7-48, six (6) of which were independent claims (i.e., claims 22, 24, 26, 28, 35, 42).

In an initial Office Action, dated September 5, 2003, claims 1-48 were rejected under 35 U.S.C. 102(e) as being anticipated by Moiin (U.S. Patent No. 6,108,699; hereinafter, Moiin). In appellants' response mailed January 5, 2004, claims 1, 3, 5, 7, 12, 17, 22, 24, 26, 28, 29, 35, 36, 42 and 43 were amended and dependent claim 49 was added.

In a second and final Office Action, dated March 16, 2004, claims 1, 3, 5, 28, 35 and 42 were rejected under 35 U.S.C. 112, first paragraph, as failing to comply with the written description requirement; claims 1-21 and 28-49 were rejected under 35 U.S.C. 102(e) as being anticipated by Moiin; and claims 22-27 were rejected under 35 U.S.C. 102(e) as being anticipated by Shrivastava et al. (U.S. Patent No. 6,449,734; hereinafter Shrivastava). In appellants' response mailed May 17, 2004, claims 1, 3, 5, 28, 35, and 42 were amended.

Appellants: Novaes et al.  
Serial No.: 09/584,259  
Filing Date: 05/31/00

Appellants received an Advisory Action, dated July 21, 2004, which indicated that appellants' amendments had been entered, but the response mailed May 17, 2004 did not place the application in condition for allowance.

A Notice of Appeal to the Board of Patent Appeals and Interferences was mailed on August 3, 2004, with the requisite fee for a one-month extension of time. The Notice of Appeal was received at the United States Patent and Trademark Office on August 6, 2004. In support of the Notice of Appeal, Appellants filed an Appeal Brief on November 1, 2004 with the United States Patent Office, along with a One-Month Extension of Time and the requisite fee.

Responsive to Appellants' Appeal Brief mailed November 1, 2004, prosecution was reopened and a new, non-final Office Action issued April 26, 2005, in which claims 1-49 were rejected under 35 U.S.C. §102(e) as being anticipated by Shrivastava. In Appellants' Response to Office Action dated July 26, 2005, claims 1-49 were canceled and new claims 50-92 were added, nine (9) of which were independent claims (i.e., claims 50, 58, 59, 63, 66, 74, 78, 86 & 90).

In a next and final Office Action dated October 19, 2005, restriction to Group I, claims 50-62 & 66-89 or Group II, claims 63-65 & 90-92, was stated, of which Group I claims 50-62 & 66-89 were provisionally elected by Appellants' representative on October 7, 2005. Additionally, claims 50-57, 66-73 & 78-85 were rejected under 35 U.S.C. §102(e) as being anticipated by Moiin et al. (U.S. Patent No. 5,999,712 A; hereinafter Moiin '712) and claims 59-62, 74-77 & 86-89 were rejected under 35 U.S.C. §103(a) as being unpatentable over Moiin '712 in view of Gamache et al. (U.S. Patent No. 6,401,120 B1; hereinafter Gamache). In Appellants' Response to Office Action dated January 24, 2006, accompanied by a One-Month Extension of Time request, claim 58 was canceled.

Appellants received an Advisory Action mailed February 14, 2006, which indicated that the request for reconsideration had been considered, but did not place the application in condition for allowance. For purposes of Appeal, the proposed amendment would be entered.

Appellants: Novaes et al.  
Serial No.: 09/584,259  
Filing Date: 05/31/00

Appellants submitted a Request for Continued Examination (RCE) and accompanying Amendment on February 21, 2006 in which claims 64, 65, 91 & 92 were canceled, and claims 50, 51, 53, 54, 59, 63, 66, 67, 69, 70, 74, 78, 79, 81, 82, 86 & 90 were amended.

In a further Office Action, dated March 29, 2006, claims 50-57, 59-63 & 66-90 were rejected under 35 U.S.C. §102(e) as being anticipated by Snaman, Jr. et al., (U.S. Patent No. 6,243,744 B1; hereinafter Snaman, Jr.). In Appellants' Response to Office Action filed July 17, 2006 with a One-Month Extension of Time, claims 50, 56, 57, 59, 66, 72-74, 78 & 80-86 were amended, and claims 54, 55, 60, 70, 71, 75, 82, 83 & 87 were canceled.

In the final Office Action mailed October 3, 2006, claims 50-53, 56-57, 59, 61-63, 66-69, 72-74, 76-81, 84-86 & 88-90 were rejected under 35 U.S.C. §102(e) as being anticipated by Snaman, Jr. In Appellants' Response to Office Action submitted December 1, 2006, no claims were amended.

Appellants received an Advisory Action, dated December 19, 2006, which indicated that Appellants' request for reconsideration had been considered but did not place the application in condition for allowance.

A Notice of Appeal to the Board of Patent Appeals and Interferences was submitted on February 2, 2007, with the requisite fee for a One-Month Extension of Time. Since the Notice of Appeal was electronically filed, the Notice was received at the United States Patent and Trademark Office on February 2, 2007.

The status of the claims is therefore as follows:

Claims allowed – none;

Claims objected to – none;

Claims rejected – 50-53, 56, 57, 59, 61-63, 66-69, 72-74, 76-81,  
84-86 & 88-90; and

Claims canceled – 1-49, 54, 55, 58, 60, 64, 65, 70, 71, 75, 82, 83,  
87, 91 & 92.

Appellants are appealing the rejection of claims 50-53, 56, 57,59, 61-63, 66-69, 72-74, 76-81, 84-86 & 88-90, as stated in the final Office Action dated October 3, 2006.

### **Status of Amendments**

Appellants proffered no amendments responsive to the final Office Action of October 3, 2006. The claims as set out in the Claims Appendix include all prior entered claim amendments.

### **Summary of Claimed Subject Matter**

In one aspect of the invention, Appellants claim a method of managing processing groups of a shared nothing distributed computing environment (e.g., *claim 50*). This method includes: requesting via a request 900 (FIG. 9A) by a prospective member 800 (FIG. 8) to join a processing group of a shared nothing distributed computing environment (FIG. 8), the request including a sequence number indicating a version of the processing group (see, e.g., page 16, line 17 – page 17, line 4); determining whether the prospective member can join the processing group, the determining employing the sequence number, wherein the determining comprises comparing 906 (FIG. 9A) by the prospective member the sequence number in the request with a current group sequence number to determine if the join of the prospective member to the processing group should continue (see, e.g., page 18, lines 1-10); joining the processing group by the prospective member, in response at least in part to the determining indicating that the prospective member can join the processing group (see, e.g., FIG. 9B and page 20, lines 12-23); and wherein the joining includes automatically reinitializing state of the prospective member responsive to the comparing indicating that the sequence number in the request is less than the current group sequence number, the reinitializing making a state of the prospective member consistent with a state of an existing member of the processing group, and thereafter, proceeding with the joining by the prospective member (see, e.g., page 18, line 11 – page 19, line 21).

In a further aspect of the invention, Appellants claim a method of managing processing groups of a shared nothing distributed computing environment (e.g., *claim 59*). This method includes: joining a prospective member 604 (FIG. 6A) to an inactive processing group of a shared nothing distributed computing environment (see, e.g., FIG. 6A, 616 (FIG. 6B), and page 14, lines 16-20); comparing a sequence number of the processing group with a sequence number of the prospective member 618 (FIG. 6B); updating the sequence number of the processing group, in response to the comparing indicating a particular difference (see, e.g., 620 (FIG. 6B), and page 14, lines 22-24); determining whether a quorum of members has joined the processing group (see, e.g., 632 (FIG. 6B), and page 14, lines 24-27); setting the sequence number of the processing group, in response to the determining indicating a quorum of members has joined the processing group; initiating activation of the processing group, in response to the setting (see, e.g., 624, 626 (FIG. 6B), and page 15, lines 1-11); and wherein the initiating activation includes obtaining by a member of the processing group having a sequence number lower than the sequence number of the processing group a copy of group state associated with the sequence number of the processing group, and reinitializing the member using the copy of group state (see, e.g., 630, 632 (FIG. 6C), and page 15, lines 12-23).

In another aspect of the present invention, Appellants recite a system of managing processing groups of a shared nothing distributed computing environment (e.g., *claim 66*). The system includes: a request 900 (FIG. 9A) by a prospective member 800 (FIG. 8) to join a processing group of a shared nothing distributed computing environment (FIG. 8), the request including a sequence number indicating a version of the processing group (see, e.g., page 16, line 17 – page 17, line 4); means for determining whether the prospective member can join the processing group, the means for determining employing the sequence number, and wherein the means for determining comprises means for comparing 906 (FIG. 9A) by the prospective member the sequence number in the request with a current group sequence number to determine if the join of the prospective member to the processing group should continue (see, e.g., page 18, lines 1-10); means for joining the processing group by the prospective member, in response at least in part to the determining indicating that the prospective member can join the processing group (see, e.g., FIG. 9B and page 20, lines 12-23); and wherein the means for joining includes

means for automatically reinitializing state of the prospective member responsive to the means for comparing indicating that the sequence number in the request is less than the current group sequence number, the reinitializing making a state of the prospective member consistent with a state of an existing member of the processing group, and thereafter, proceeding with the joining by the prospective member (see, e.g., page 18, line 11 – page 19, line 21).

In a further aspect of the invention, Appellants recite a system of managing processing groups of a shared nothing distributed computing environment (e.g., *claim 74*). The system includes: means for joining a prospective member 604 (FIG. 6A) to an inactive processing group of a shared nothing distributed computing environment (see, e.g., FIG. 6A, 616 (FIG. 6B), and page 14, lines 16-20); means for comparing a sequence number of the processing group with a sequence number of the prospective member 618 (FIG. 6B); means for updating the sequence number of the processing group, in response to the comparing indicating a particular difference (see, e.g., 620 (FIG. 6B), and page 14, lines 22-24); means for determining whether a quorum of members has joined the processing group (see, e.g., 632 (FIG. 6B), and page 14, lines 24-27); means for setting the sequence number of the processing group, in response to the determining indicating a quorum of members has joined the processing group; means for initiating activation of the processing group, in response to the setting (see, e.g., 624, 626 (FIG. 6B), and page 15, lines 1-11); and wherein the means for initiating activation includes means for obtaining by a member of the processing group having a sequence number lower than the sequence number of the processing group a copy of group state associated with the sequence number of the processing group, and means for reinitializing the member using the copy of group state (see, e.g., 630, 632 (FIG. 6C), and page 15, lines 12-23).

In a further aspect of the present invention, Appellants claim an article of manufacture (e.g., *claim 78*). The article of manufacture includes at least one computer-usable medium having computer-readable program code logic (see, e.g., page 36, line 17 – page 37, line 9) to manage processing groups of a shared nothing distributed computing environment. The computer-readable program code logic includes: a request 900 (FIG. 9A) by a prospective member 800 (FIG. 8) to join a processing group of a shared nothing distributed computing environment (FIG. 8), the request including a sequence number indicating a version of the

processing group (see, e.g., page 16, line 17 – page 17, line 4); determine logic to determine whether the prospective member can join the processing group, the determining employing the sequence number, wherein the determining comprises comparing 906 (FIG. 9A) by the prospective member the sequence number in the request with a current group sequence number to determine if the join of the prospective member to the processing group should continue (see, e.g., page 18, lines 1-10); join logic to join the processing group by the prospective member, in response at least in part to the determining indicating that the prospective member can join the processing group (see, e.g., FIG. 9B and page 20, lines 12-23); and wherein the join logic includes automatic reinitialize logic to automatically reinitialize state of the prospective member responsive to the comparing indicating that the sequence number in the request is less than the current group sequence number, the automatic reinitialize logic making a state of the prospective member consistent with the state of an existing member of the processing group, and thereafter, proceeding with the joining by the prospective member (see, e.g., page 18, line 11 – page 19, line 21).

In a further aspect of the present invention, Appellants claim an article of manufacture (e.g., *claim 86*). This article of manufacture includes at least one computer-usable medium having computer-readable program code logic (see, e.g., page 36, line 17 – page 37, line 9) to manage processing groups of a shared nothing distributed computing environment. The computer-readable program code logic includes: join logic to join a prospective member 604 (FIG. 6A) to an inactive processing group of a shared nothing distributed computing environment (see, e.g., FIG. 6A, 616 (FIG. 6B), and page 14, lines 16-20); compare logic to compare a sequence number of the processing group with a sequence number of the prospective member 618 (FIG. 6B); update logic to update the sequence number of the processing group, in response to the comparing indicating a particular difference (see, e.g., 620 (FIG. 6B), and page 14, lines 22-24); determine logic to determine whether a quorum of members has joined the processing group (see, e.g., 632 (FIG. 6B), and page 14, lines 24-27); set logic to set the sequence number of the processing group, in response to the determining indicating a quorum of members has joined the processing group; initiate logic to initiate activation of the processing group, in response to the setting (see, e.g., 624, 626 (FIG. 6B), and page 15, lines 1-11); and



wherein the initiate logic includes obtain logic to obtain by a member of the processing group having a sequence number lower than the sequence number of the processing group a copy of group state associated with the sequence number of the processing group, and reinitialize logic to reinitialize the member using the copy of group state (see, e.g., 630, 632 (FIG. 6C), and page 15, lines 12-23).

In another aspect, dependent *claims 52, 68 & 80* further recite quiescing activity 902 (FIG. 9A) that may effect the state prior to updating the state associated with the processing group, in response to the request, wherein the updating provides the current group sequence number (see, e.g., FIG. 10, and page 17, lines 5-17).

**Grounds of Rejection to Be Reviewed On Appeal**

1. Whether claims 50-53, 56, 57, 59, 61-63, 66-69, 72-74, 76-81, 84-86 & 88-90 were anticipated under 35 U.S.C. §102(e) to one of ordinary skill in the art by Snaman, Jr.

### Argument

#### I. Rejection under 35 U.S.C. §102(e) Over U.S. Patent No. 6,243,744 B1 (to Snaman, Jr.)

##### A. Claims 50, 51, 53, 56, 57, 59, 61-63, 66, 67, 69, 72-74, 76-79, 81, 84-86 & 88-90

Reversal of the rejection to claims 50, 51, 53, 56, 57, 59, 61-63, 66, 67, 69, 72-74, 76-79, 81, 84-86 & 88-90 as anticipated by Snaman, Jr. is respectfully requested.

##### Shared Nothing:

The present invention relates to a processing protocol for managing processing groups of a *shared nothing* distributed computing environment. A shared nothing distributed computing environment is a particular computing environment architecture that is well known to one of ordinary skill in the art. By way of example, an internet search for a *shared nothing system* uncovers many references to the particular architecture at issue. Evidence Appendix A and Evidence Appendix B are two examples of parallel hardware architecture descriptions of a shared nothing system. As noted in these materials, a shared nothing system is a particular system architecture wherein only one processing node is connected to a given disk, with the *nodes being interconnected via a network only*. This particular architecture is referred to as *shared nothing* since there is no shared disc between two more processing nodes of the system. A *shared nothing* computing environment is a distinct computing architecture from a *shared disk* architecture.

##### Shared Disk:

A *shared disk* system refers to a computing architecture wherein one or more disks are shared by two or more CPUs or nodes in the system. In a shared disk system, the two or more nodes sharing the one or more disks have access to the same disks. Central to the shared disk system is the concept that one or more disks are shared by two or more nodes of the system.

In the instant case, Appellants' recited invention comprises protocol for a shared nothing distributed computing environment, while the applied patent (Snaman, Jr.) presents protocol for a shared disk system. This difference is significant, and based thereon, Appellants respectfully submit that there is no anticipation of the recited invention based upon the teachings of Snaman, Jr.

Snaman, Jr. describes a technique for sharing a resource among a cluster of devices in a computer network. As shown in FIGS. 1A-1E & 4A-4G, Snaman, Jr. discloses a computing environment wherein a *quorum disk* is employed in forming a cluster by voting devices attempting to reach a quorum. The *quorum disk* described by Snaman, Jr. is a non-processing voting device that is *shared by at least two nodes of the environment*. Because the quorum disk is shared, the computing environment of Snaman, Jr. is a *shared disk* computing environment, not a shared nothing computing environment, as recited in Appellants' independent claims.

In this regard, Appellants respectfully traverse the characterizations contained at pages 12 & 13 of the final Office Action regarding the meaning of a *shared nothing* computing environment. These conclusions are separately addressed below.

At page 12, lines 19 & 20 of the final Office Action, it is stated that:

In response to A. above, it is the teaching of the "quorum disk" that in fact teaches that the system of Snaman, Jr. is a shared nothing computing environment.

As noted above, the quorum disk in Snaman, Jr. is a *shared disk* that is taught to be shared by nodes A & B of the system. This shared disk is shown in FIGS. 1A-1E, as well as FIGS. 4A-4G. In each of these figures, two nodes connect to the quorum disk by dedicated lines 20, 22 (in FIGS. 1A-1E) or dedicated lines 100, 102 (in FIGS. 4A-4G). Because nodes A & B share the quorum disk in Snaman, Jr., Snaman, Jr. presents a *shared disk* system. As noted above, a *shared nothing* computing environment means that there is not a shared device or disk between the nodes of the environment. This interpretation of Snaman, Jr. and Appellants' independent claims is believed well supported by the art, as represented by Evidence Appendices A & B attached herewith. The difference in architecture is significant since the protocols employed in the architectures are vastly different.

At page 12, line 20 – page 13, line 3 of the final Office Action, it is stated:

It is well known in the art that a shared nothing computing environment is a computing environment wherein applications cannot be distributed across multiple nodes.

This conclusion is believed a mischaracterization of the term “shared nothing computing environment”. The term “shared nothing computing environment” refers to a particular hardware architecture, as explained above. Further, it is incorrect to state that applications cannot be distributed across multiple nodes in a shared nothing computing environment since they can. In a shared nothing computing environment, the only connection between nodes is the network itself. However, there are existing protocols (such as MPI or DB2) which allow applications to be distributed across multiple nodes in a shared nothing distributed computing environment.

At page 13, lines 2-5 of the final Office Action, it is stated:

As such, the reason that it is possible for a node to take over running an application when the active node fails in a shared nothing computing environment, is because all the nodes in the cluster are connected to a shared storage mechanism (quorum disk).

As noted, a shared nothing computing environment refers to a particular computing environment architecture wherein there is no shared device or disk between two or more nodes of the environment. Appellants recite a protocol for managing processing groups within such a shared nothing computing environment (i.e., an environment where there is no shared quorum disk).

At page 13, lines 5-6 of the final Office Action, it is stated that:

In fact, quorums are only used in a shared nothing environment.

Appellants respectfully traverse this statement. As taught by Snaman, Jr., quorum is employed in a shared disk system as well. The difference is in the protocol which employs the quorum. In Snaman, Jr., the quorum disk controls which nodes are granted quorum, while in a shared nothing computing environment, quorum is decided among the nodes using a protocol such as recited by Appellants.

At page 13, lines 7-10 of the final Office Action, it is stated:

Quorum is necessary in a shared nothing clustering environment so that it is able to tell which node is active and which node or nodes are in stand by (to maintain consistent functionality), and also so that when failure occurs, only the partition that owns the quorum remains running the application (to maintain data consistency).

Appellants agree with this particular statement; however, note that in a shared nothing computing environment architecture, the nodes can only count on communication via the network, whereas in a shared disk computing architecture such as taught by Snaman, Jr., at least some of the nodes are also in direct communication via the shared disk. In the Snaman, Jr. protocol, the shared disk is employed as part of the protocol for responding to a failure.

At page 13, lines 10-12 of the final Office Action, it is stated that:

The fact that all the nodes share a quorum disk does not teach away from a shared nothing computing environment but rather teaches of a shared nothing computing environment.

This statement is clearly contradictory to the well established meaning of *shared nothing* computing environment versus a *shared disk* computing environment. A shared disk computing environment means that there is a shared device (i.e., a disk) between two or more nodes of the computing environment. This shared disk plays a significant role in overseeing the establishing of a quorum subsequent to failure of a node in a shared disk computing environment. In contrast, the *shared nothing* computing environment has no shared device or shared disk between two more nodes of the computing environment. The only connection between nodes in a shared nothing computing environment is the network itself.

In addition to the above, Appellants respectfully submit that the protocol of Snaman, Jr. would be inoperable in a shared nothing computing environment since the shared disk protocol described therein relies upon the existence of the disk itself. A shared quorum disk protocol such as described by Snaman, Jr. pre-existed Appellants' invention and is directed to a different computing architecture than Appellants' recited invention. In a shared nothing computing environment, there is typically no physical way to connect a shared device, for example, between remote clusters of nodes. A shared disk environment such as described by Snaman, Jr. requires dedicated connections. These special hardware connections couple two or more nodes of the system to the shared disk, as illustrated in all of the Snaman, Jr. architecture figures. The Snaman, Jr. shared disk computing environment is referred to in the art as a "twin-tailed" disk configuration, wherein two nodes of the system connect to the shared disk. This is not the environment recited by Appellants in the independent claims presented. In Appellants' independent claims, a shared nothing computing environment is expressly recited.

Given this characterization, Appellants respectfully submit that there is no anticipation of their recited invention based upon the teachings of Snaman, Jr., nor would Appellants' invention have been obvious to one of ordinary skill in the art based upon the teachings of Snaman, Jr. The protocol described by Snaman, Jr. expressly requires the existence of the shared disk. Further, one skilled in the art would not have modified the teachings of Snaman, Jr. so as to remove the shared quorum disk, since Snaman, Jr. relies on the existence of the shared device for the clustering protocol described therein. The quorum disk in Snaman, Jr. is critical to situations in which no cluster has a majority, and the quorum disk provides the configuration information. Thus, once a processing node arrives at the quorum disk, the configuration can be obtained from the quorum disk. Without the quorum disk, the Snaman, Jr. protocol would be unworkable.

In contrast, Appellants' recited protocol does not rely upon nor implement shared disk protocol. In Appellants' recited invention, the environment is a *shared nothing* computing environment, meaning that there is not a shared device between the nodes of the environment (as represented by the Evidence Appendices A & B material, as well as the understanding of one of ordinary skill in the art).

For at least the above-noted reasons, Appellants respectfully submit that Appellants' claims patentably distinguish over the teachings of Snaman, Jr. Accordingly, reversal of the 35 U.S.C. §102(e) rejection of claims 50, 51, 53, 56, 57, 59, 61-63, 66, 67, 69, 72-74, 76-79, 81, 84-86 & 88-90 based on Snaman, Jr. is respectfully requested.

B. Claims 52, 68 & 80:

Reversal of the rejection to claims 52, 68 & 80 as anticipated by Snaman, Jr. is also respectfully requested.

Dependent claims 52, 68 & 80 recite quiescing activity that may effect the state prior to the updating. The term "quiescing" is a term of art which refers to the gradual quieting of a system or network. Quiescing occurs when a signal is sent to stop processing, and then a period of time passes while processes (gracefully) discontinue processing. The processes are essentially allowed to save their current state and exit voluntarily. In Snaman, Jr., column 13, lines 36-39, it

is noted that Snaman, Jr. presents a method and apparatus for stopping a partitioned cluster such that processing operations are not performed which would destroy cluster information on a shared resource. As explained at column 12, line 62+, Snaman, Jr. stops by crashing the cluster to remove any possibility of performing data processing operations that would destroy cluster information on the shared resource. This crashing of the cluster is clearly not an analogous to Appellants' recited quiescing activity that may effect state prior to the updating. The two are distinct functions, and one does not suggest the other. Because Snaman, Jr. is a quorum disk based approach, Snaman, Jr. cannot risk data corruption, and the cluster must crash. In contrast, Appellants' invention is not quorum disk based (i.e., is a shared nothing based protocol), and the processes are allowed to quiesce voluntarily.

For at least the above-noted additional reasons, Appellants respectfully request reversal of the 35 U.S.C. §102(e) rejection of claims 52, 68 & 80 based on Snaman, Jr.

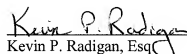
### **Conclusion**

Appellants request reversal of the §102(e) rejections to claims 50-53, 56, 57, 59, 61-63, 66-69, 72-74, 76-81, 84-86 & 88-90 based on Snaman, Jr. Appellants respectfully submit that this document would not have rendered their claimed invention anticipated to one of ordinary skill in the art. Snaman, Jr. does not teach or suggest at least Appellants' recited protocol for a shared nothing distributed computing environment. The protocol described by Snaman, Jr. requires the existence of a shared quorum disk. One skilled in the art would not have modified the teachings of Snaman, Jr. to remove the shared quorum disk, since Snaman, Jr. relies on the shared disk for the clustering protocol described therein.

Appellants: Novaes et al.  
Serial No.: 09/584,259  
Filing Date: 05/31/00

For all of the above reasons, Appellants allege error in rejecting their claims as anticipated over the applied art. Accordingly, reversal of the rejection is respectfully requested.

Respectfully submitted,

  
Kevin P. Radigan, Esq.  
Attorney for Appellants  
Registration No.: 31,789

Dated: March 30, 2007

HESLIN ROTHENBERG FARLEY & MESITI P.C.  
5 Columbia Circle  
Albany, New York 12203-5160  
Telephone: (518) 452-5600  
Facsimile: (518) 452-5579



**Claims Appendix**

1-49. (Cancelled).

50. A method of managing processing groups of a shared nothing distributed computing environment, said method comprising:

requesting via a request by a prospective member to join a processing group of a shared nothing distributed computing environment, said request including a sequence number indicating a version of the processing group;

determining whether the prospective member can join the processing group, said determining employing the sequence number, wherein the determining comprises comparing by said prospective member the sequence number in the request with a current group sequence number to determine if the join of the prospective member to the processing group should continue;

joining the processing group by the prospective member, in response at least in part to the determining indicating that the prospective member can join the processing group; and

wherein the joining comprises automatically reinitializing state of the prospective member responsive to the comparing indicating that the sequence number in the request is less than the current group sequence number, the reinitializing making a state of the prospective member consistent with a state of an existing member of the processing group, and thereafter, proceeding with the joining by the prospective member.

51. The method of claim 50, further comprising updating state associated with the processing group, in response to the request, said updating providing the current group sequence number.

52. The method of claim 51, further comprising quiescing activity that may affect the state prior to said updating.

53. The method of claim 50, wherein the determining specifies that the join should continue if the compare indicates that the sequence number in the request is less than the current group sequence number, otherwise the join should not continue.

54. (Canceled).

55. (Canceled).

56. The method of claim 50, further comprising determining an activity status of the processing group prior to the reinitializing, wherein the reinitializing is performed if the processing group is active.

57. The method of claim 50, wherein the joining further comprises updating the current group sequence number.

58. (Cancelled).

59. A method of managing processing groups of a shared nothing distributed computing environment, said method comprising:

joining a prospective member to an inactive processing group of a shared nothing distributed computing environment;

comparing a sequence number of the processing group with a sequence number of the prospective member;

updating the sequence number of the processing group, in response to the comparing indicating a particular difference;

determining whether a quorum of members has joined the processing group;

setting the sequence number of the processing group, in response to the determining indicating a quorum of members has joined the processing group;

initiating activation of the processing group, in response to the setting; and

wherein the initiating activation comprises:

obtaining by a member of the processing group having a sequence number lower than the sequence number of the processing group a copy of group state associated with the sequence number of the processing group; and

reinitializing the member using the copy of group state.

60. (Canceled).

61. The method of claim 59, wherein activation of the processing group comprises updating the sequence number of the processing group.

62. The method of claim 61, wherein the updating of the sequence number of the processing group comprises updating the sequence number, in response to there being a majority of members in the processing group.

63. The method of claim 59, wherein a member comprises a distributed synchronous transaction system.

64. (Cancelled).

65. (Cancelled).

66. A system of managing processing groups of a shared nothing distributed computing environment, said system comprising:

a request by a prospective member to join a processing group of a shared nothing distributed computing environment, said request including a sequence number indicating a version of the processing group;

means for determining whether the prospective member can join the processing group, said means for determining employing the sequence number, and wherein the means for determining comprises means for comparing by said prospective member the sequence number in the request with a current group sequence number to determine if the join of the prospective member to the processing group should continue;

means for joining the processing group by the prospective member, in response at least in part to the determining indicating that the prospective member can join the processing group; and

wherein the means for joining comprises means for automatically reinitializing state of the prospective member responsive to the means for comparing indicating that the sequence number in the request is less than the current group sequence member, the reinitializing making a state of the prospective member consistent with a state of an existing member of the processing group, and thereafter, proceeding with the joining by the prospective member.

67. The system of claim 66, further comprising means for updating state associated with the processing group, in response to the request, said updating providing the current group sequence number.

68. The system of claim 67, further comprising means for quiescing activity that may affect the state prior to said updating.

69. The system of claim 66, wherein the determining specifies that the join should continue if the compare indicates that the sequence number in the request is less than the current group sequence number, otherwise the join should not continue.

70. (Canceled).

71. (Canceled).

72. The system of claim 66, further comprising means for determining an activity status of the processing group prior to the reinitializing, wherein the reinitializing is performed if the processing group is active.

73. The system of claim 66, wherein the means for joining further comprises means for updating the current group sequence number.

74. A system of managing processing groups of a shared nothing distributed computing environment, said system comprising:

means for joining a prospective member to an inactive processing group of a shared nothing distributed computing environment;

means for comparing a sequence number of the processing group with a sequence number of the prospective member;

means for updating the sequence number of the processing group, in response to the comparing indicating a particular difference;

means for determining whether a quorum of members has joined the processing group;

means for setting the sequence number of the processing group, in response to the determining indicating a quorum of members has joined the processing group;

means for initiating activation of the processing group, in response to the setting; and

wherein the means for initiating activation comprises:

means for obtaining by a member of the processing group having a sequence number lower than the sequence number of the processing group a copy of group state associated with the sequence number of the processing group; and

means for reinitializing the member using the copy of group state.

75. (Canceled).

76. The system of claim 74, wherein activation of the processing group comprises means for updating the sequence number of the processing group.

77. The system of claim 76, wherein the means for updating of the sequence number of the processing group comprises means for updating the sequence number, in response to there being a majority of members in the processing group.

78. An article of manufacture comprising:

at least one computer usable medium having computer readable program code logic to manage processing groups of a shared nothing distributed computing environment, the computer readable program code logic comprising:

a request by a prospective member to join a processing group of a shared nothing distributed computing environment, said request including a sequence number indicating a version of the processing group;

determine logic to determine whether the prospective member can join the processing group, said determining employing the sequence number, wherein the determining comprises comparing by said

prospective member the sequence number in the request with a current group sequence number to determine if the join of the prospective member to the processing group should continue;

join logic to join the processing group by the prospective member, in response at least in part to the determining indicating that the prospective member can join the processing group; and

wherein the join logic comprises automatic reinitialize logic to automatically reinitialize state of the prospective member responsive to the comparing indicating that the sequence number in the request is less than the current group sequence number, the automatic reinitialize logic making a state of the prospective member consistent with the state of an existing member of the processing group, and thereafter, proceeding with the joining by the prospective member.

79. The article of manufacture of claim 78, further comprising update logic to update state associated with the processing group, in response to the request, said updating providing the current group sequence number.

80. The article of manufacture of claim 79, further comprising quiesce logic to quiesce activity that may affect the state prior to said updating.

81. The article of manufacture of claim 78, wherein the determining specifies that the join should continue if the compare indicates that the sequence number in the request is less than the current group sequence number, otherwise the join should not continue.

82. (Canceled).

83. (Canceled).

84. The article of manufacture of claim 78, further comprising determine logic to determine an activity status of the processing group prior to the reinitializing, wherein the reinitializing is performed if the processing group is active.

85. The article of manufacture of claim 78, wherein the join logic further comprises update logic to update the current group sequence number.

86. An article of manufacture comprising:

at least one computer usable medium having computer readable program code logic to manage processing groups of a shared nothing distributed computing environment, the computer readable program code logic comprising:

join logic to join a prospective member to an inactive processing group of a shared nothing distributed computing environment;

compare logic to compare a sequence number of the processing group with a sequence number of the prospective member;

update logic to update the sequence number of the processing group, in response to the comparing indicating a particular difference;

determine logic to determine whether a quorum of members has joined the processing group;

set logic to set the sequence number of the processing group, in response to the determining indicating a quorum of members has joined the processing group;

initiate logic to initiate activation of the processing group, in response to the setting; and

wherein the initiate logic comprises:



obtain logic to obtain by a member of the processing group having a sequence number lower than the sequence number of the processing group a copy of group state associated with the sequence number of the processing group; and

reinitialize logic to reinitialize the member using the copy of group state.

87. (Canceled).

88. The article of manufacture of claim 86, wherein activation of the processing group comprises update logic to update the sequence number of the processing group.

89. The article of manufacture of claim 88, wherein the update logic to update the sequence number of the processing group comprises update logic to update the sequence number, in response to there being a majority of members in the processing group.

90. The article of manufacture of claim 86, wherein a member comprises a distributed synchronous transaction system.

91. (Cancelled).

92. (Cancelled).

\* \* \* \* \*

Appellants: Novaes et al.  
Serial No.: 09/584,259  
Filing Date: 05/31/00

**Evidence Appendix**

Attached hereto are Evidence Appendix A and Evidence Appendix B, which are two examples of parallel hardware architecture descriptions of a shared nothing system. These materials were included with Appellants' Response to Office Action submitted December 1, 2006, requesting reconsideration of the final Office Action mailed October 3, 2006. As indicated in the Advisory Action dated December 19, 2006, Appellants' Response to Office Action was entered for purposes of appeal.

## EVIDENCE APPENDIX A

Parallel Hardware Architecture

Page 1 of 9

### Oracle7 Parallel Server Concepts and Administrator's Guide



## Parallel Hardware Architecture

- Overview
- [Required Hardware and Operating System Software](#)
- [Shared Memory Systems](#)
- [Shared Disk Systems](#)
- [Shared Nothing Systems](#)
- [Shared Nothing/Shared Disk Combined Systems](#)

The parallel database server can use various machine architectures which allow parallel processing. This chapter describes the range of available hardware implementations and surveys their advantages and disadvantages.

- [Overview](#)
- [Required Hardware and Operating System Software](#)
- [Shared Memory Systems](#)
- [Shared Disk Systems](#)
- [Shared Nothing Systems](#)
- [Shared Nothing/Shared Disk Combined Systems](#)

### Overview

This section covers the following topics:

- [Parallel Processing Hardware Implementations](#)
- [Application Profiles](#)

Oracle configurations support parallel processing within a machine, between machines, and between nodes. There is no advantage to running Oracle Parallel Server on a single node and a single system image—you would incur overhead and receive no benefit. With standard Oracle you do not have to do anything special on shared memory configurations to take advantage of some parallel processing capabilities.

Although this manual focuses on Oracle Parallel Server with shared nothing/shared disk architecture, the application design issues discussed in this book may also be relevant to standard Oracle systems.

#### Parallel Processing Hardware Implementations

<http://thinkunix.net/unix/db/oracle/docs-7.3/DOC/server/doc/SPS73/chap3.htm>

11/29/2006

Parallel processing hardware implementations are often categorized according to the particular resources which are shared. The following categories are described in this chapter:

- shared memory systems
- shared disk systems
- shared nothing systems

These implementations can also be described as "tightly coupled" or "loosely coupled," according to the way in which communication between nodes is accomplished:

Implementation Type	Tightly Coupled	Loosely Coupled
Shared Memory	SMP	NUMA
Shared Disk	MPP	Clusters
Shared Nothing	MPP	Clusters

Table 3 - 1. Classification of Parallel Processing Hardware Implementations

**Attention:** Oracle supports *all* these different implementations of parallel processing.

The following table shows interrelations between various hardware architectures and available Oracle options.

System Configuration	Shared Memory	Shared Disk	Shared Nothing
Type of Oracle	standard Oracle	Oracle Parallel Server	Oracle Parallel Server
Parallel Query Option	available	available	available

Table 3 - 2. Hardware Architecture and Oracle Solutions

**Note:** Support for any given Oracle configuration is platform-dependent; check to confirm that your platform supports the configuration you want.

The preceding table assumes that in a shared nothing system the software enables a node to access a disk from another node. For example, the IBM SP2 features a virtual shared disk: the disk is shared through software.

### Application Profiles

Online transaction processing (OLTP) applications tend to perform best on symmetric multiprocessors or clusters. Decision support (DSS) applications tend to perform well on massively parallel systems. Application profiles and parallel processing hardware implementations can typically be represented as follows:

Implementation Type	OLTP	DSS
Shared Memory (SMP)	Good	Good
Shared Disk (Cluster)	OK if carefully implemented	Good
Shared Nothing (MPP)	Not optimal	Good

Table 3 - 3. Parallel Processing Implementations and Application Profiles

Choose the implementation that provides the power you need for the application(s) you require.

## Required Hardware and Operating System Software

Each hardware vendor implements parallel processing in its own way, but the following common elements are required for Oracle Parallel Server:

- High Speed Interconnect
- Globally Accessible Disk or Shared Disk Subsystem
- Distributed Lock Manager

### **High Speed Interconnect**

This is a high bandwidth, low latency communication facility between the various nodes for lock manager and cluster manager traffic. The interconnect can be Ethernet, FDDI, or some other proprietary interconnect method. If the primary interconnect fails, a back-up interconnect is usually available. The back-up interconnect will ensure high availability, and prevent a single point of failure.

### **Globally Accessible Disk or Shared Disk Subsystem**

All nodes in a loosely coupled or massively parallel system have simultaneous access to shared disks. This gives multiple instances of Oracle7 concurrent access to the same database. These shared disk subsystems are most often implemented via a shared SCSI or twintailed SCSI (common in UNIX) connected to a disk farm. On some MPP platforms, such as IBM SP, disks are associated to nodes and a virtual shared disk software layer enables global access to all nodes.

### **Distributed Lock Manager**

The distributed lock manager (DLM) consists of software and hardware that coordinates resource sharing in a loosely coupled or massively parallel system. The distributed lock manager tracks "ownership" of a resource, accepts requests for resources from processes, notifies requesting processes when a resource is available, and grants shared or exclusive access to a resource for a process.

Oracle uses the DLM to coordinate modifications of data blocks, maintenance of cache consistency, recovery of failed nodes, transaction locks, dictionary locks, and SCN locks.

See Also: "Distributed Lock Manager: Access to Resources" ■

## **Shared Memory Systems**

This section describes:

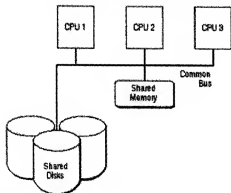
- Tightly Coupled Systems
- Uniform and Non-Uniform Memory Access
- Summary: Shared Memory Systems

### **Tightly Coupled Systems**

Tightly coupled shared memory systems, illustrated in Figure 3 - 1, have the following characteristics:

- Multiple CPUs share memory.
- Each CPU has full access to all shared memory through a common bus.

- Communication between nodes occurs via shared memory.
- Performance is limited by the bandwidth of the memory bus.



Shared Memory System

Figure 3 - 1. Tightly Coupled

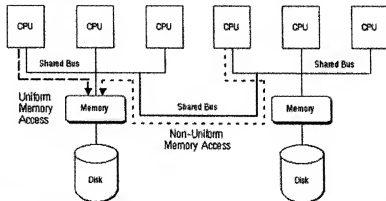
Symmetric multiprocessor (SMP) machines are often nodes in a cluster. Multiple SMP nodes can be used with Oracle Parallel Server in a tightly coupled system, where memory is shared among the multiple CPUs, and is accessible by all the CPUs through a memory bus. Examples of tightly coupled systems include the Pyramid, Sequent, and Sun SparcServer.

It does not make sense to run Oracle Parallel Server on a single SMP machine, because the system would incur a great deal of unnecessary overhead from DLM accesses.

Performance is potentially limited in a tightly coupled system by a number of factors. These include various system components such as the memory bandwidth, CPU to CPU communication bandwidth, the memory available on the system, the I/O bandwidth, and the bandwidth of the common bus.

#### Uniform and Non-Uniform Memory Access

Shared memory systems can be loosely coupled with memory. Figure 3 - 2 shows two ways in which shared memory can be accessed: uniform memory access from the CPU on the left, and non-uniform memory access (NUMA) between the left and right disks.



Uniform Memory Access

Figure 3 - 2. Uniform and Non-

The Oracle Parallel Server can work with either form of memory access—but NUMA is a more expensive form of memory access and synchronization than uniform memory access. While any CPU can access the memory, it is more costly for the remote nodes to do this.

### **Summary: Shared Memory Systems**

Parallel processing advantages of shared memory systems are described in this section.

#### **Advantages**

- Memory access is cheaper than inter-node communication. This means that internal synchronization is faster than using the distributed lock manager.
- Shared memory systems are easier to administer than a cluster.

## **Shared Disk Systems**

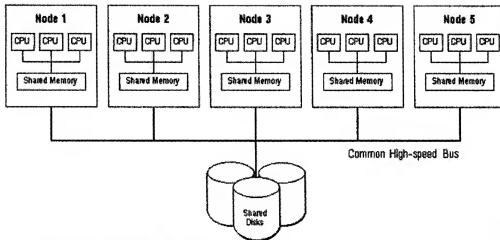
Shared disk systems are typically loosely coupled. This section describes:

- Loosely Coupled Systems
- Summary: Shared Disk Systems

### **Loosely Coupled Systems**

Loosely coupled shared disk systems, illustrated in Figure 3-3, have the following characteristics:

- Each node consists of one or more CPUs and associated memory.
- Memory is not shared between nodes.
- Communication occurs over a common high-speed bus.
- Each node has access to the same disks and other resources.
- A node can be an SMP if the hardware supports it. For example, nCUBE does not support an SMP, but many loosely coupled UNIX systems do support SMPs.
- Bandwidth of the high-speed bus limits the number of nodes (scalability) of the system.



**Figure 3 - 3. Loosely Coupled Shared Disk System**

The cluster illustrated in Figure 3 - 3 is composed of multiple tightly coupled nodes. A DLM is required. Examples of loosely coupled systems are VAXclusters or Sun clusters.

Since the memory is not shared among the nodes, each node has its own data cache. Cache consistency must be maintained across the nodes and a lock manager is needed to maintain the consistency. Additionally, instance locks using the DLM on the Oracle level must be maintained to ensure that all nodes in the cluster see identical data.

There is additional overhead in maintaining the locks and ensuring that the data caches are consistent. The performance impact is dependent on the hardware and software components, such as the bandwidth of the high-speed bus through which the nodes communicate, and DLM performance.

#### **Summary: Shared Disk Systems**

Parallel processing advantages and disadvantages of shared disk systems are described in this section.

##### **Advantages**

- Shared disk systems permit high availability. All data is accessible even if one node dies.
- These systems have the concept of one database, which is an advantage over shared nothing systems.
- Shared disk systems provide for incremental growth.

##### **Disadvantages**

- Inter-node synchronization is required, involving DLM overhead and greater dependency on high-speed interconnect.
- If the workload is not partitioned well, there may be high synchronization overhead.
- There is operating system overhead of running shared disk software.

#### **Shared Nothing Systems**

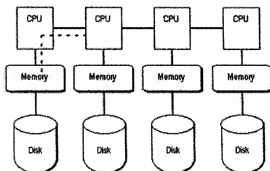


Shared nothing systems are typically loosely coupled. This section describes:

- [Overview of Shared Nothing Systems](#)
- [Massively Parallel Systems](#)
- [Summary: Shared Nothing Systems](#)

### Overview of Shared Nothing Systems

In shared nothing systems only one CPU is connected to a given disk. If a table or database is located on that disk, access depends entirely on the CPU which owns it. If the CPU fails the data cannot be accessed--regardless of how many other CPUs may still be running. Shared nothing systems can be represented as follows:



System

Figure 3 - 4. Shared Nothing

Shared nothing systems are concerned with access to disks, not access to memory. Oracle Parallel Server can access the disks on a shared nothing system as long as the operating system provides transparent disk access, but this access is expensive in terms of latency.

Shared nothing systems are fundamentally concerned with access to disks, not memory. Nonetheless, adding more CPUs and disks can improve scaleup.

### Massively Parallel Systems

Massively parallel (MPP) systems, illustrated in [Figure 3 - 5](#), have the following characteristics:

- From only a few nodes, up to thousands of nodes are supported.
- The cost per processor may be extremely low because each node is an inexpensive processor.
- Each node has associated non-shared memory.
- Each node has its own devices, but in case of failure other nodes can access the devices of the failed node (on most systems).
- Nodes are organized in a grid, mesh, or hypercube arrangement.
- Oracle instances can potentially reside on any or all nodes.

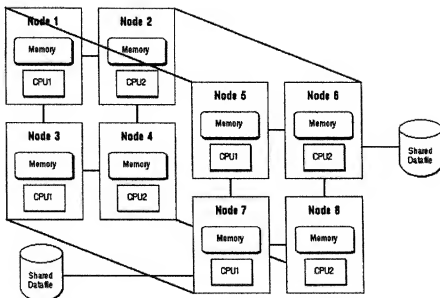


Figure 3 - 5. Massively Parallel System: A Hypercube Example

Note: A hypercube is an arrangement of processors such that each processor is connected to  $\log_2 n$  other processors, where  $n$  is the number of processors in the hypercube.  $\log_2 n$  is said to be the "dimension" of the hypercube. For example, in the 8-processor hypercube shown in Figure 3 - 5, dimension = 3; each processor is connected to three other processors.

A massively parallel system may have as many as several thousand nodes. Each node may have its own Oracle instance, with all the standard facilities of an instance. (An Oracle instance comprises the System Global Area and all the background processes.)

An MPP has access to a huge amount of real memory for all database operations (such as sorts or the buffer cache), since each node has its own associated memory. To avoid disk I/O, this advantage will be significant in long running queries and sorts. This is not possible for 32 bit machines which have a 2 GB addressing limit; the total amount of memory on an MPP system may well be over 2 GB.

As with loosely coupled systems, cache consistency on MPPs must still be maintained across all nodes in the system. Thus, the overhead for cache management is still present.

Examples of massively parallel systems are the nCUBE2 Scalar Supercomputer, the Unisys OPUS, Amdahl, Meiko, and the IBM SP.

### Summary: Shared Nothing Systems

Parallel processing advantages and disadvantages of shared nothing systems are described in this section.

#### Advantages

- Shared nothing systems provide for incremental growth.
- System growth is practically unlimited.

- MPPs are good for read-only databases and decision support applications.
- Failure is local: if one node fails, the others stay up.

#### Disadvantages

- More coordination is required.
- A process can only work on the node that owns the desired disk.
- If one node dies, processes cannot access its data.
- Physically separate databases which are logically one database can be extremely complex and time-consuming to administer.
- Adding nodes means reconfiguring and taping out data on disks.
- If there is a heavy workload of updates or inserts, as in an online transaction processing system, it may be worthwhile to consider data-dependent routing to alleviate contention.

### Shared Nothing /Shared Disk Combined Systems

A combined system can be very advantageous—one which brings together the advantages of shared nothing and shared disk, while overcoming their respective limitations. Such a combined system can be represented as follows:

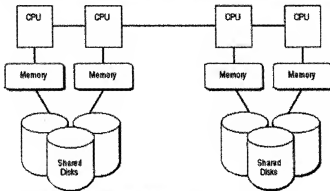


Figure 3 - 6. Two Shared Disk

#### Systems Forming a Shared Nothing System

Here, two shared disk systems are linked to form a system with the same hardware redundancies as a shared nothing system. If one CPU fails, the other CPUs can still access all disks.



ORACLE  
Copyright © 1996 Oracle Corporation.  
All Rights Reserved.



## EVIDENCE APPENDIX B

Parallel Hardware Architecture

Page 1 of 8

**Oracle8i Parallel Server Concepts and Administration**  
**Release 8.1.5**  
A67778-01



### 3 Parallel Hardware Architecture

You can deploy Oracle Parallel Server (OPS) on various architectures. This chapter describes hardware implementations that accommodate the parallel server and explains their advantages and disadvantages.

- [Overview](#)
- [Required Hardware and Operating System Software](#)
- [Shared Memory Systems](#)
- [Shared Disk Systems](#)
- [Shared Nothing Systems](#)
- [Shared Nothing /Shared Disk Combined Systems](#)

#### Overview

This section covers the following topics:

- [Parallel Processing Hardware Implementations](#)
- [Application Profiles](#)

Oracle configurations support parallel processing within a machine, between machines, and between nodes. There is no advantage to running OPS on a single node with a single instance; you would incur overhead without receiving benefits. With standard Oracle you do not have to do anything special on shared memory configurations to take advantage of some parallel processing capabilities.

Although this manual focuses on OPS on a shared nothing/shared disk architecture, the application design issues discussed in this book may also be relevant to standard Oracle systems.

#### Parallel Processing Hardware Implementations

We often categorize parallel processing hardware implementations according to the particular resources that are shared. This chapter describes these categories:

[http://www.csee.umbc.edu/help/oracle8/server.815/a67778/ch3\\_arch.htm](http://www.csee.umbc.edu/help/oracle8/server.815/a67778/ch3_arch.htm)

11/29/2006

- Shared memory systems
- Shared disk systems
- Shared nothing systems

These implementations can also be described as "tightly coupled" or "loosely coupled", according to the way the nodes communicate.

Oracle supports *all* these implementations of parallel processing, assuming that in a shared nothing system the software enables a node to access a disk from another node. For example, the IBM SP2 features a virtual shared disk: the disk is shared through software.

**Note:**

Support for any given Oracle configuration is platform-dependent; check whether your platform supports your desired configuration.

## Application Profiles

Online transaction processing (OLTP) applications tend to perform best on symmetric multiprocessors; they perform well on clusters and MPP systems if they can be well partitioned. Decision support (DSS) applications tend to perform well on SMFs, clusters, and massively parallel systems. Select the implementation providing the power you need for the application(s) you require.

## Required Hardware and Operating System Software

Each hardware vendor implements parallel processing in its own way, but the following common elements are required for OPS:

- High Speed Interconnect
- Globally Accessible Disk or Shared Disk Subsystem

### High Speed Interconnect

This is a high bandwidth, low latency communication facility among nodes for lock manager and cluster manager traffic. The interconnect can be Ethernet, FDDI, or some other proprietary interconnect method. If the primary interconnect fails, a back-up interconnect is usually available. The back-up interconnect ensures high availability, and prevents single points of failure.

### Globally Accessible Disk or Shared Disk Subsystem

All nodes in loosely coupled or massively parallel systems have simultaneous access to shared disks. This gives multiple instances of Oracle8 concurrent access to the same database. These shared disk subsystems are most often implemented by way of shared SCSI or twin-tailed SCSI (common in UNIX) connections to a disk farm. On some MPP platforms, such as IBM SP, disks are associated to nodes and

a virtual shared disk software layer enables global access to all nodes.

---

**Note:**

The Integrated Distributed Lock Manager (IDLM) coordinates modifications of data blocks, maintenance of cache consistency, recovery of failed nodes, transaction locks, dictionary locks, and SCN locks.

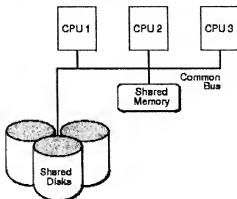
---

## Shared Memory Systems

Tightly coupled shared memory systems, illustrated in Figure 3-1, have the following characteristics:

- Multiple CPUs share memory
- Each CPU has full access to all shared memory through a common bus
- Communication among nodes occurs by way of shared memory
- Performance is limited by memory bus bandwidth

**Figure 3-1 Tightly Coupled Shared Memory System**



Symmetric multiprocessor (SMP) machines are often comprised of nodes in a cluster. You can install multiple SMP nodes with OPS in a tightly coupled system where memory is shared among the multiple CPUs, and is accessible by all the CPUs through a memory bus. Examples of tightly coupled systems include the Pyramid, Sequent, and Sun SparcServer.

It does not make sense to run OPS on a single SMP machine, because the system would incur a great deal of unnecessary overhead from IDLM accesses.

Performance is potentially limited in a tightly coupled system by a number of factors. These include

various system components such as the memory bandwidth, CPU-to-CPU communication bandwidth, the memory available on the system, the I/O bandwidth, and the common bus bandwidth.

Parallel processing advantages of shared memory systems are these:

- Memory access is less expensive than inter-node communication: this means internal synchronization is faster than using the Lock Manager
- Shared memory systems are easier to administer than a cluster

A disadvantage of shared memory systems for parallel processing is:

- Scalability is limited by bus bandwidth and latency, and by available memory

## Shared Disk Systems

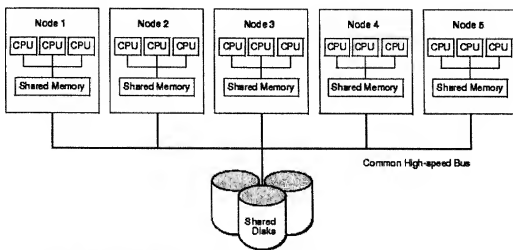
Shared disk systems are typically loosely coupled. Such systems, illustrated in Figure 3-2, have the following characteristics:

- Each node consists of one or more CPUs and associated memory
- Memory is not shared among nodes
- Communication occurs over a common high-speed bus
- Each node has access to the same disks and other resources
- A node can be an SMP if the hardware supports it
- Bandwidth of the high-speed bus limits the number of nodes (scalability) of the system

***Figure 3-2 Loosely Coupled Shared Disk System***

#### Parallel Hardware Architecture

Page 5 of 8



The cluster illustrated in Figure 3-2 is composed of multiple, tightly coupled nodes. The IDLM is required. Examples of loosely coupled systems are VAX clusters or Sun clusters.

Since memory is not shared among the nodes, each node has its own data cache. Cache consistency must be maintained across the nodes and a lock manager is needed to maintain the consistency. Additionally, instance locks using the IDLM on the Oracle level must be maintained to ensure all nodes in the cluster see identical data.

There is additional overhead in maintaining the locks and ensuring data cache consistency. The effect on performance is dependent on the hardware and software components, such as the high-speed bus bandwidth through which the nodes communicate, and IDLM performance.

Parallel processing advantages of shared disk systems are:

- Shared disk systems permit high availability. All data is accessible even if one node dies.
- These systems have the concept of "one database", which is an advantage over shared nothing systems.
- Shared disk systems provide for incremental growth.

Parallel processing disadvantages of shared disk systems are:

- Inter-node synchronization is required, involving IDLM overhead and greater dependency on high-speed interconnect.
- If the workload is not partitioned well, there may be high synchronization overhead.
- There is operating system overhead of running shared disk software.

---

**Note:**

[http://www.csce.umbc.edu/help/oracle8/server.815/a67778/ch3\\_arch.htm](http://www.csce.umbc.edu/help/oracle8/server.815/a67778/ch3_arch.htm)

11/29/2006



Memory mapped hardware available in late 1998 will provide functionality to copy buffers directly from one user address on one node to another user address on another node.

## Shared Nothing Systems

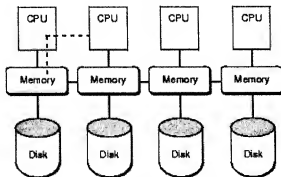
Shared nothing systems are typically loosely coupled. This section describes:

- Overview of Shared Nothing Systems
- Massively Parallel Systems
- Summary of Shared Nothing Systems

### Overview of Shared Nothing Systems

In shared nothing systems, only one CPU is connected to a given disk. If a table or database is located on that disk, access depends entirely on the CPU that owns it. Figure 3-3 illustrates shared nothing systems:

**Figure 3-3 Shared Nothing System**



Shared nothing systems are concerned with access to disks, not access to memory. Nonetheless, adding more CPUs and disks can improve scaleup. OPS can access the disks on a shared nothing system as long as the operating system provides transparent disk access, but this access is expensive in terms of latency.

### Massively Parallel Systems

Massively parallel (MPP) systems have these characteristics:

- MPP systems can range in size from only a few nodes to up to thousands of nodes
- The cost per processor may be extremely low because each node is an inexpensive processor

- Each node has associated non-shared memory
- Each node may have its own devices, but during failures other nodes can access the devices of the failed node
- Nodes are organized in a grid, mesh, or hypercube arrangement
- Oracle instances can potentially reside on any or all nodes

As mentioned, an MPP system can have as many as several thousand nodes. Each node may have its own Oracle instance with all the standard facilities of an instance. (An Oracle instance comprises the System Global Area and all the background processes.)

An MPP has access to a huge amount of real memory for all database operations (such as sorts or the buffer cache), since each node has its own associated memory. To avoid disk I/O, this advantage is important to long running queries and sorts. This is not possible for 32-bit machines which have 2GB addressing limits; total memory on MPP systems may be over 2GB. As with loosely coupled systems, cache consistency on MPPs must still be maintained across all nodes in the system. Thus, the overhead for cache management is still present. Examples of MPP systems are the nCUBE2 Scalar Supercomputer, the Unisys OPUS, Amdahl, Meiko, Pyramid, Smile, and the IBM SP.

### Summary of Shared Nothing Systems

Shared nothing systems have advantages and disadvantages for parallel processing:

#### Advantages

- Shared nothing systems provide incremental growth
- System growth is practically unlimited
- MPPs are generally good for read-only, DSS applications
- Failure is local: if one node fails, the others stay up

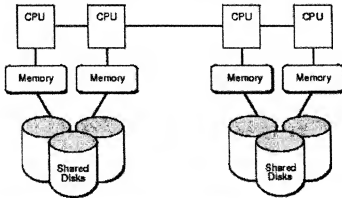
#### Disadvantages

- More coordination is required
- More overhead is required for processes working on a disk belonging to another node
- If there is a heavy workload of updates or inserts, as in online transaction processing systems, it may be worthwhile to consider data-dependent routing to reduce contention.

### Shared Nothing /Shared Disk Combined Systems

A combined system can be very advantageous. This unites advantages of shared nothing and shared disk, while overcoming their respective limitations. Figure 3-4 illustrates a combined system:

**Figure 3-4 Two Shared Disk Systems Forming a Shared Nothing System**



Here, two shared disk systems are linked to form a system with the same hardware redundancies as a shared nothing system. If one CPU fails, the other CPUs can still access all disks.



ORACLE  
Copyright © 1999 Oracle Corporation.  
All Rights Reserved.



Appellants: Novaes et al.  
Serial No.: 09/584,259  
Filing Date: 05/31/00

**Related Proceedings Appendix**

None.